CSE 344

APRIL 11TH – DATALOG

ADMINISTRATIVE MINUTIAE

- **HW2 Due tonight**
- **HW3 out this afternoon**
- **OQ4 Out**
- **Midterm**
  - Fill out piazza quiz before tomorrow

DATALOG: FACTS AND RULES

Facts = tuples in the database

Rules = queries

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

← Schema

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

```
Q2(f, l) :-  Actor(z,f,l), Casts(z,x),
             Movie(x,y,'1940').
```

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                         Casts(z,x2), Movie(x2,y2,1940)
```

Extensional Database Predicates = EDB = Actor, Casts, Movie

Intensional Database Predicates = IDB = Q1, Q2, Q3

ParentChild(p,c)

SAFE DATALOG RULES

Here are *unsafe* datalog rules. What's "unsafe" about them ?

```
U1(x,y) :- ParentChild("Alice",x), y != "Bob"
```

```
U2(x)   :- ParentChild("Alice",x), !ParentChild(x,y)
```

ParentChild(p,c)

SAFE DATALOG RULES

Here are *unsafe* datalog rules. What's "unsafe" about them ?

U1(x,y) :- ParentChild("Alice",x), y != "Bob"

Holds for
every y other than "Bob"
U1 = infinite!

U2(x) :- ParentChild("Alice",x), !ParentChild(x,y)

ParentChild(p,c)

SAFE DATALOG RULES

Here are *unsafe* datalog rules. What's "unsafe" about them ?

U1(x,y) :- ParentChild("Alice",x), y != "Bob"

Holds for every y other than "Bob" U1 = infinite!

U2(x) :- ParentChild("Alice",x), !ParentChild(x,y)

Want Alice's childless children, but we get all children x (because there exists some y that x is not parent of y)

ParentChild(p,c)

SAFE DATALOG RULES

Here are *unsafe* datalog rules. What's "unsafe" about them ?

U1(x,y) :- ParentChild("Alice",x), y != "Bob"

Holds for every y other than "Bob" U1 = infinite!

U2(x) :- ParentChild("Alice",x), !ParentChild(x,y)

Want Alice's childless children, but we get all children x (because there exists some y that x is not parent of y)

A datalog rule is *safe* if every variable appears in some positive relational atom

DATALOG: RELATIONAL DATABASE

- **Datalog can express things RA cannot**
  - Recursive Queries
- **Can Datalog express all queries in RA?**

RELATIONAL ALGEBRA OPERATORS

Union $\cup$, difference **-**

Selection σ

Projection π

Cartesian product **X**, join $\bowtie$

OPERATORS IN DATALOG

- Suppose we want Q1(…) to contain all the values from F1(…) and F2(…)

OPERATORS IN DATALOG

- **Suppose we want Q1(…) to contain all the values from F1(…) and F2(…)**
  - Q1(…) :- F1(…)
  - Q1(…) :- F2(…)
- **What about for difference?**

OPERATORS IN DATALOG

- **Suppose we want Q1(…) to contain all the values from F1(…) and F2(…)**
  - Q1(…) :- F1(…)
  - Q1(…) :- F2(…)
- **What about for difference?**
  - Q1(…) :- F1(…), !F2(…)

OPERATORS IN DATALOG

- **Suppose we want Q1(…) to contain all the values from F1(…) and F2(…)**
  - Q1(…) :- F1(…)
  - Q1(…) :- F2(…)
- **What about for difference?**
  - Q1(…) :- F1(…), !F2(…)
  - The variables (…) in F1 and F2 must be the same, or else we have an *unsafe* rule

OPERATORS IN DATALOG

- **Projection, from the variables R_1,R_2,...R_k select some subset of the variables**

OPERATORS IN DATALOG

- **Projection, from the variables R_1,R_2,...R_k select some subset of the variables**
  - Q1(subset) :- Original(all_attributes)
- **Selection: only return certain records from our knowledge base**

OPERATORS IN DATALOG

- **Projection, from the variables R_1,R_2,...R_k select some subset of the variables**
  - Q1(subset) :- Original(all_attributes)
- **Selection: only return certain records from our knowledge base**
  - Q1(…) :- Original(…), selection_criteria

OPERATORS IN DATALOG

- Cross product: find all the pairs between $R(a_1, a_2 \ldots)$ and $S(b_1, b_2 \ldots)$

OPERATORS IN DATALOG

- **Cross product: find all the pairs between $R(a_1,a_2\ldots)$ and $S(b_1,b_2\ldots)$**
  - $Q1(a_1,b_1,a_2,b_2\ldots)$:- $R(a_1,a_2\ldots)$, $S(b_1,b_2\ldots)$
- **Joins?**
  - Natural

OPERATORS IN DATALOG

- **Cross product: find all the pairs between $R(a_1,a_2\ldots)$ and $S(b_1,b_2\ldots)$**
  - $Q1(a_1,b_1,a_2,b_2\ldots) \text{ :- } R(a_1,a_2\ldots), S(b_1,b_2\ldots)$
- **Joins?**
  - Natural: Q1(a,b,c) :- R(a,b), S(b,c)
  - Theta

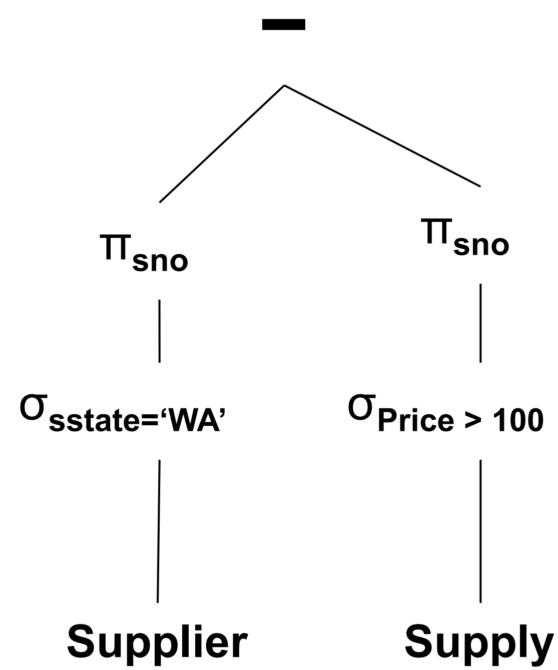
OPERATORS IN DATALOG

- **Cross product: find all the pairs between $R(a_1,a_2\ldots)$ and $S(b_1,b_2\ldots)$**
  - $Q1(a_1,b_1,a_2,b_2\ldots)$:- $R(a_1,a_2\ldots)$, $S(b_1,b_2\ldots)$
- **Joins?**
  - Natural: Q1(a,b,c) :- R(a,b), S(b,c)
  - Theta: Cross product with selection
  - Equijoin: subset of Theta join

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)

EXAMPLE

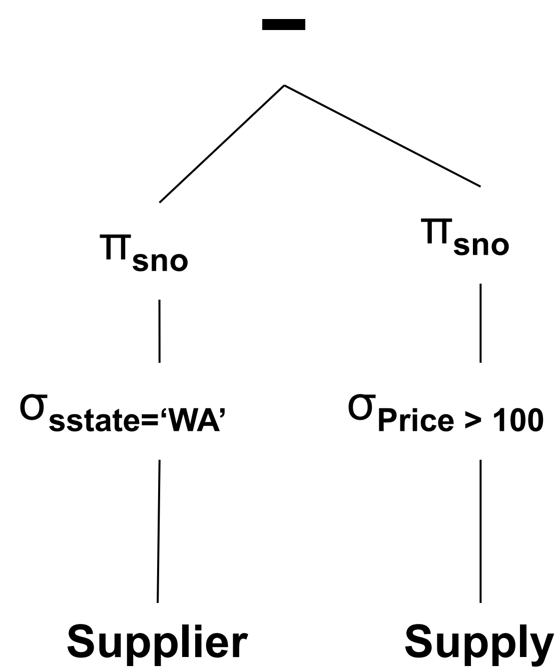
```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = ‘WA’)
    EXCEPT
 (SELECT P.sno
   FROM Supply P
   WHERE P.price > 100)
```



EXAMPLE

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

Datalog:

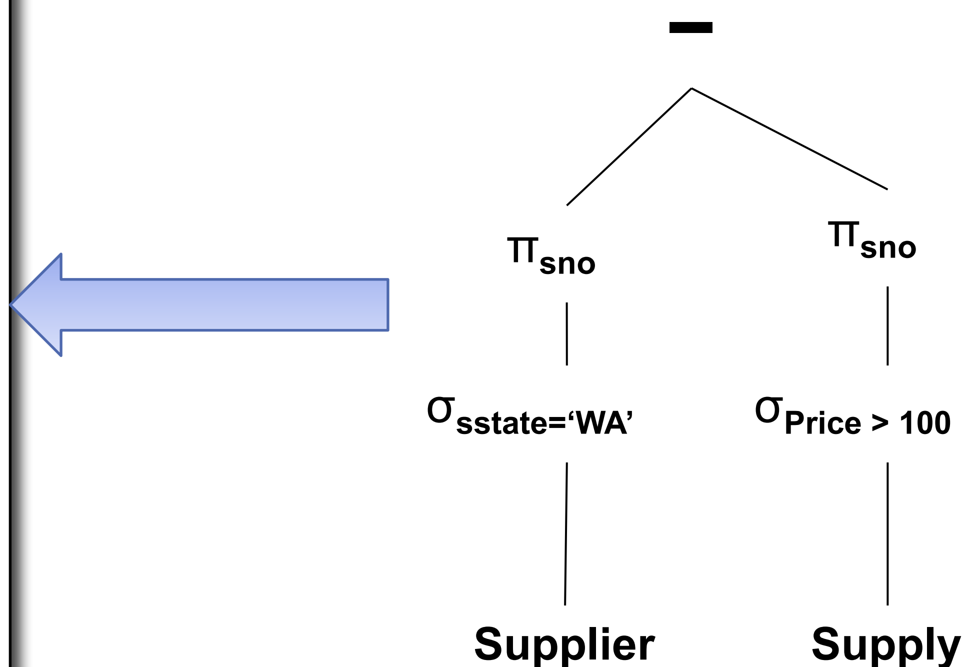


Supplier(<u>sno</u>,sname,scity,sstate)
Part(<u>pno</u>,pname,psize,pcolor)
Supply(<u>sno,pno</u>,price)

EXAMPLE

```
Datalog:

Q1(no,name,city,state) :-
    Supplier(sno,sname,
    scity,sstate),
    sstate='WA'
Q2(no,pno,price) :-
    Supply(s,pn,pr),
    pr > 100
Q3(sno) :- Q1(sno,n,c,s)
Q4(sno) :- Q2(sno,pn,pr)
Result(sno) :- Q3(sno),
    !Q4(sno)
```



Friend(name1, name2)
Enemy(name1, name2)

MORE EXAMPLES W/O RECURSION

Find Joe's friends, and Joe's friends of friends.

```
A(x) :- Friend('Joe', x)
A(x) :- Friend('Joe', z), Friend(z, x)
```

MORE EXAMPLES W/O RECURSION

Find all of Joe's friends who do not have any friends except for Joe:

```
JoeFriends(x) :- Friend('Joe',x)
NonAns(x) :- JoeFriends(x), Friend(x,y), y != ‘Joe’
A(x) :- JoeFriends(x), NOT NonAns(x)
```

MORE EXAMPLES W/O RECURSION

Find all people such that all their enemies' enemies are their friends

Q: if someone doesn't have any enemies nor friends, do we want them in the answer?

A: Yes!

```
Everyone(x) :- Friend(x,y)
Everyone(x) :- Friend(y,x)
Everyone(x) :- Enemy(x,y)
Everyone(x) :- Enemy(y,x)
NonAns(x) :- Enemy(x,y),Enemy(y,z), NOT Friend(x,z)
A(x) :- Everyone(x), NOT NonAns(x)
```

MORE EXAMPLES W/O RECURSION

Find all persons x that have a friend all of whose enemies are x's enemies.

```
Everyone(x) :- Friend(x,y)
NonAns(x) :- Friend(x,y) Enemy(y,z), NOT Enemy(x,z)
A(x) :- Everyone(x), NOT NonAns(x)
```

ParentChild(p,c)

MORE EXAMPLES W/ RECURSION

Two people are in the *same generation* if they are siblings, or if they have parents in the same generation

Find all persons in the same generation with Alice

ParentChild(p,c)

MORE EXAMPLES W/ RECURSION

Find all persons in the same generation with Alice

Let's compute SG(x,y) = "x,y are in the same generation"

```
SG(x,y) :- ParentChild(p,x), ParentChild(p,y)
SG(x,y) :- ParentChild(p,x), ParentChild(q,y), SG(p,q)
Answer(x) :- SG("Alice", x)
```

DATALOG SUMMARY

EDB (base relations) and IDB (derived relations)

Datalog program = set of rules

Datalog is recursive

Some reminders about semantics:

- Multiple atoms in a rule mean join (or intersection)
- Variables with the same name are join variables
- Multiple rules with same head mean union

CLASS OVERVIEW

Unit 1: Intro

Unit 2: Relational Data Models and Query Languages

Unit 3: Non-relational data

- NoSQL
- Json
- SQL++

Unit 4: RDMBS internals and query optimization

Unit 5: Parallel query processing

Unit 6: DBMS usability, conceptual design

Unit 7: Transactions

Unit 8: Advanced topics (time permitting)